Using Active Testing and Meta-Level Information for Selection of Classification Algorithms

Pavel Brazdil
Rui Leite
Joaquin Vanschoren
Francisco Queiros
Report CW 591, August 2010

Katholieke Universiteit Leuven
Department of Computer Science
Celestijnenlaan 200A – B-3001 Heverlee (Belgium)

Using Active Testing and Meta-Level Information for Selection of Classification Algorithms

Pavel Brazdil
Rui Leite
Joaquin Vanschoren
Francisco Queiros
Report CW 591, August 2010

Department of Computer Science, K.U.Leuven

Abstract

The problem of selecting the best classification algorithm for a specific problem continues to be very relevant, especially since the number of classification algorithms keeps growing significantly. Testing all alternatives is not really a viable option: if we compare all pairs of algorithms, as is often advocated, the number of comparisons grows exponentially. To avoid this problem we suggest a method referred to as *active testing*, whose aim is to reduce the number of comparisons by carefully selecting which tests should be carried out.

This method uses meta-knowledge concerning past experiments and proceeds in an iterative manner. It takes the form of a competition in which, in each iteration, the candidate best algorithm is pitted against its *most promising competitor*. The winner proceeds to the next round, while the loser is removed from consideration. To speed up the process of testing each pair of competitors on new datasets, we use a fast method that exploits meta-information on partial learning curves measured on prior datasets to predict which algorithm is better. We stop when there are no more viable competitors.

This method was evaluated in a leave-one-out fashion, and results show that it is indeed effective in determining the best algorithm using a limited number of tests.

Keywords : machine learning, meta-learning, active testing
MSC : Primary : I.2.6, Secondary : I.5.0, I.5.2.

Using Active Testing and Meta-Level Information for Selection of Classification Algorithms

Pavel Brazdil*, Rui Leite*, Joaquin Vanschoren† and Francisco Queiros‡
*LIAAD-INESC Porto L.A./Faculty of Economics, University of Porto, Portugal,
Email: pbrazdil@liaad.up.pt, rleite@liaad.up.pt
†Department of Computer Science, Katholieke Universiteit Leuven, Belgium,
Email: joaquin.vanschoren@cs.kuleuven.be
‡Faculty of Economics, University of Porto, Portugal,
Email: 080401175@fep.up.pt

***Abstract*—The problem of selecting the best classification algorithm for a specific problem continues to be very relevant, especially since the number of classification algorithms keeps growing significantly. Testing all alternatives is not really a viable option: if we compare all pairs of algorithms, as is often advocated, the number of comparisons grows exponentially. To avoid this problem we suggest a method referred to as *active testing*, whose aim is to reduce the number of comparisons by carefully selecting which tests should be carried out.**

This method uses meta-knowledge concerning past experiments and proceeds in an iterative manner. It takes the form of a competition in which, in each iteration, the candidate best algorithm is pitted against its *most promising competitor*. The winner proceeds to the next round, while the loser is removed from consideration. To speed up the process of testing each pair of competitors on new datasets, we use a fast method that exploits meta-information on partial learning curves measured on prior datasets to predict which algorithm is better. We stop when there are no more viable competitors.

This method was evaluated in a leave-one-out fashion, and results show that it is indeed effective in determining the best algorithm using a limited number of tests.

***Keywords*-active learning; metalearning; algorithm selection;**

ACKNOWLEDGMENT

Acknowledgements to be added to camera ready version upon acceptance.

I. INTRODUCTION

The problem of selecting the best classification algorithm for a specific problem continues to be very relevant. The number of classification algorithms has grown significantly. Moreover, the performance of some classification algorithms is very sensitive to parameter settings. Therefore the issue of algorithm selection (or in general KDD workflows) has been an object of study in many works during the past 20 years [1], [2], [3], [4], [5]. Most approaches rely on the concept of *metalearning*. This approach exploits information concerning dataset characteristics and the performance of algorithm on prior datasets to guide the search for the best algorithm(s) to be used on the current dataset. The term *metalearning* stems from the fact that we try to learn the function that maps *problem / dataset characterizations* to *algorithm performance estimates*, which is then applied to a new problem / dataset. The dataset characterization is often captured in the form of various statistical and information-theoretic measures [1], [2].

Others have advocated the use *sampling landmarks*, representing performance results of algorithms on subsets of data [6], [7]. These can be used to characterize the given datasets as other measures do. A series of sampling landmarks represents in effect a partial learning curve. In a subsequent work [8], [9] it was shown that use of sampling landmarks leads to substantially better results than previous approaches that exploit just classical dataset characteristics. Moreover, it was shown that it is possible to combine *sampling landmarks* with classical dataset characteristics, benefitting from the information in both types of meta-data [9].

However, it has been argued that the characterization of classification algorithms is sensitive to which pair of algorithms we have in mind [10]. This is also true when considering sampling landmarks. Therefore, some authors have focused on the particular subtask of selecting the most appropriate algorithm from a pair. Extending this to the case of multiple algorithms, some authors have proposed that a set of pairwise statistical tests be carried out. Then, some aggregate measure (such as the overall number of "wins") is used to identify the best candidate, or alternatively, construct a ranking of candidate algorithms [10], [11]. However, the disadvantage of this approach is that the number of binary tests grows exponentially with the number of algorithms. To tackle this problem we suggest a method, referred to as *active testing*, which bears some similarities to *active learning*. The aim is to reduce the number of necessary comparisons by carefully selecting which tests should be carried out.

This method uses meta-knowledge concerning past experiments and proceeds in an iterative manner. It takes the form of a competition in which, in each iteration, the candidate best algorithm is pitted against its *most promising competitor*. The winner proceeds to the next round, while

the loser is removed from consideration. To speed up the process of testing each pair of competitors on new datasets, we use a fast method that exploits information on partial learning curves [9] to predict which algorithm is better. We stop when there are no more alternatives to be explored.

An added benefit is that by combining the tests generated by this method over different datasets, we can build a single decision tree which features the most important pairwise tests in the nodes. It provides an effective plan of tests to be carried out in a new situation (i.e. for a new dataset).

The paper is organized as follows. In Section II we discuss the underlying assumptions of this work. In Section III we outline the main steps of the proposed method and describe each of these steps in detail. Section IV evaluates this method, both in a small-scale and large-scale setting including hundreds of algorithm-parameter settings. The final section presents discussion, future work and conclusions.

II. Underlying Assumptions

In this section we describe some underlying assumptions of this work. In the following sections, we test these assumptions by implementing and evaluating the method.

A. Meta-data: Probability Estimates

Given a set of classification algorithms with given parameter settings and some new classification problem (dataset d_{new}) the aim is to predict the best algorithm for this task with respect to some given performance measure (e.g. predictive accuracy or AUC). Each algorithm-parameter setting combination is handled as a different algorithm. Let us represent the performance of algorithm a_i on dataset d_{new} as $M(a_i, d_{new})$. Our aim is then to predict the algorithm $a*$ with the highest performance evaluation on d_{new}. Hence, the algorithm $a*$ should satisfy the following constraint:

$$\forall a_i M(a*, d_{new}) \geq M(a_i, d_{new}) \tag{1}$$

As our estimate $\hat{a*}$ might not be perfect, we are interested in the probability

$$P(M(\hat{a*}, d_{new}) \geq M(a_i, d_{new})) \tag{2}$$

We want this probability to be as high as possible, ideally near to 1. The decision concerning $\geq$ may be established using a statistical significance test or even using a simple comparison.

As in other work that exploits metalearning, we will assume that the probability estimate, or at least the relative performance of two algorithms, can be predicted based on similar past cases. Let Ds represent a subset of all datasets D that is similar to d_{new} (later on we will explain how such similar datasets can be identified). The assumption that is being explored here is the following:

$$P(M(\hat{a*}, d_{new}) \geq M(a_i, d_{new})) \sim P(M(\hat{a*}, Ds) \geq M(a_i, Ds))$$

That is, the subset of datasets Ds is used to estimate this probability. For a fixed Ds and a particular a_i (e.g. the best current candidate) the latter estimate can be done quite easily. We simply go though all algorithms (except a_i) and identify the one that satisfies the constraint in the majority of cases. As such, the objective is:

$$\frac{\arg\max_{a_k} \sum_{dj \in Ds} (i(M(a_k, d_j) \geq M(a_i, d_j)))}{|Ds|} \tag{3}$$

where $|Ds|$ represents the size of Ds and $i(test)$ an identity function whose value is 1, if the $test$ is true and 0 otherwise.

B. Dataset Similarity

Let us now see how the subset of dataset Ds can be identified for a given d_{new}. This could be done with the help of various statistical and information-theoretic measures that were mentioned earlier [1], [2]. Another possibility is to use algorithm evaluations already carried out on the new dataset. Suppose we have carried out a comparison with a_1 and a_2 and obtained, say, the result $M(a_2, d_{new}) \geq M(a_1, d_{new})$. The result of this test can then be used as a dataset characteristic. If this situation has occurred on another dataset (i.e. a_2 achieved a better result than a_1, in short $a_2 > a_1$), the two datasets can be said to be similar in this respect, if not, it can said to be dissimilar. It is assumed that this notion is useful to identify the best algorithm for new datasets.

C. Active Learning vs. Active Testing

The proposed active testing approach bears some similarity to active learning methods. In *active learning* [21], e.g. for supervised classification [13], one aims to avoid the expensive labeling of all examples by intelligently selecting which of the unlabeled data point is most informative should the class be known, and should be labeled next. This can be based, for instance, on the entropy in the data distribution or the uncertainty of the predictions for the unlabeled examples. With active testing, we aim to avoid performing all pairwise tests by intelligently selecting which algorithms should be compared first, based on their relative performance on similar datasets. The assumption adopted here is that if the probability of a_j winning over a_i on d_{new} is high then this test is *informative* and is a good candidate test to be carried out.

D. Other Work on Active Learning

Quite a lot of work has been done on experiment design [12] and active learning. Our method described here follows the main trends that have been outlined in literature. There is relatively little work on active learning for ranking tasks.

One notable exception is [14], who use the notion of Expected Loss Optimization (ELO). Another work that is relevant is [15]. Their aim was to identify the best substances for drug screening using a minimum number of tests. In the experiments described, the authors have focused on top-10 substances. Several different strategies were considered and evaluated. Our problem here is not ranking, but rather finding the potentially the best element, so this work is only partially relevant.

III. Active Testing

In this section we describe a method, called AT for Active Testing, that is based on the assumptions discussed in the previous section. Its aim is to identify the best classification algorithm for a given dataset in a minimal number of pairwise tests. The method involves the following steps:

1) Choose a good initial candidate algorithm (see Section III-A). While this step is not crucial (a random algorithm would also work), it can save several pairwise tests.
2) Find the *most promising competitor* algorithm, based on algorithm meta-data: the relative performances of the two candidate algorithms on prior datasets (see Section III-B).
3) Perform a pairwise test of the two algorithms on the new dataset. This could be a cross-validated evaluation or a (faster) meta-learning prediction based on partial learning curves measured on prior datasets (see Section III-C).
4) Remove the losing algorithm from consideration and mark the winning algorithm as the new candidate algorithm. Repeat the whole process starting with step 2 until no promising competitor remains.

The following subsections present more details concerning each step.

A. An Initial Algorithm Ranking

The first step of our AT procedure is to select a good initial candidate algorithm. This can simply be the algorithm performing best overall on prior datasets. Alternatively, we could select this initial candidate using any meta-learning technique based purely on data characteristics ([1], [2]), but this outside the scope of this paper.

Here, it is assumed that performance evaluations of the algorithms on prior datasets are available. This performance information can be expressed by *success rate (predictive accuracy)* or its corresponding rank, *AUC* (area under the ROC curve), or any other suitable measure of classifier performance. Here we use *predictive accuracy* and the corresponding ranks.

To facilitate the explanation of the AT procedure, we start with a small-scale (toy) study. It involves 6 different classification algorithms (with default parameter settings) from Weka [16]: 1-Nearest Neighbor (IB1), C4.5 (J48), Ripper (JRip), Logistic discriminant (LogD), Artificial neural nets (MLP) and naive Bayes (NB), on 40 UCI datasets [17].

Table I
Ranking of Algorithms and Mean Rank

Datasets	IB1		J48		JRip		LogD		MLP		NB	
abalone	.197	(5)	.218	(4)	.185	(6)	.259	(2)	.266	(1)	.237	(3)
acetylation	.844	(1)	.831	(2)	.829	(3)	.745	(5)	.609	(6)	.822	(4)
adult_metal	.794	(6)	.861	(1)	.843	(3)	.850	(2)	.830	(5)	.834	(4)
allbp	.964	(5)	.973	(2)	.975	(1)	.966	(4)	.970	(3)	.942	(6)
allhyper	.975	(5)	.988	(1)	.985	(2)	.979	(4)	.981	(3)	.954	(6)
ann	.924	(6)	.997	(1)	.995	(2)	.959	(4)	.971	(3)	.954	(5)
byzantine	.990	(2)	.952	(5)	.936	(6)	.986	(3)	.991	(1)	.980	(4)
car	.778	(6)	.931	(3)	.882	(4)	.935	(2)	.992	(1)	.855	(5)
cmc	.428	(6)	.527	(2)	.523	(3)	.506	(4)	.532	(1)	.498	(5)
contraceptive	.428	(6)	.527	(2)	.523	(3)	.506	(4)	.532	(1)	.498	(5)
injury_severity	.770	(6)	.839	(2)	.827	(3)	.813	(4)	.845	(1)	.771	(5)
internetad	.960	(4)	.967	(2)	.968	(1)	.951	(5)	NA	(6)	.966	(3)
isolet	.897	(2)	.841	(4)	.787	(5)	NA	(6)	.970	(1)	.851	(3)
krkopt	.372	(5)	.564	(2)	.395	(4)	.405	(3)	.635	(1)	.360	(6)
krvskp	.900	(5)	.994	(2)	.991	(3)	.976	(4)	.994	(1)	.877	(6)
led24	.482	(6)	.716	(3)	.701	(4)	.725	(1)	.681	(5)	.724	(2)
letter	.959	(1)	.881	(2)	.859	(3)	.774	(5)	.827	(4)	.642	(6)
mfeat	.978	(2)	.940	(4)	.920	(5)	NA	(6)	.984	(1)	.953	(3)
musk	.957	(4)	.970	(2)	.961	(3)	.952	(5)	1.000	(1)	.838	(6)
nursery	.776	(6)	.970	(2)	.967	(3)	.925	(4)	.998	(1)	.902	(5)
optdigits	.988	(1)	.902	(6)	.904	(5)	.939	(3)	.984	(2)	.914	(4)
page	.958	(5)	.968	(2)	.970	(1)	.965	(3)	.961	(4)	.898	(6)
parity	.508	(3)	.779	(2)	.449	(4)	.371	(5)	.922	(1)	.368	(6)
pendigits	.994	(1)	.962	(3)	.963	(2)	.956	(4)	.945	(5)	.857	(6)
pyrimidines	.946	(3)	.946	(2)	.936	(4)	.907	(5)	.959	(1)	.822	(6)
quadrupeds	1.000	(3)	1.000	(6)	1.000	(3)	1.000	(3)	1.000	(3)	1.000	(3)
quisclas	.577	(5)	.622	(3)	.644	(2)	.694	(1)	.578	(4)	.396	(6)
recljan2jun97	.942	(5)	.946	(4)	.948	(3)	.949	(2)	.949	(1)	.850	(6)
sat	.901	(1)	.861	(4)	.863	(3)	.858	(5)	.893	(2)	.795	(6)
segmentation	.974	(1)	.969	(2)	.960	(3)	.957	(5)	.958	(4)	.800	(6)
shuttle	.999	(3)	1.000	(1)	1.000	(2)	.968	(5)	.997	(4)	.930	(6)
sick	.961	(5)	.986	(1)	.985	(2)	.969	(4)	.970	(3)	.928	(6)
spambase	.907	(4)	.932	(1)	.926	(3)	.927	(2)	.895	(5)	.797	(6)
splice	.749	(6)	.941	(3)	.932	(4)	.907	(5)	.956	(1)	.954	(2)
taska_part_hhold	.533	(4)	.803	(1)	.801	(2)	NA	(5.5)	NA	(5.5)	.578	(3)
thyroid0387	.817	(4)	.958	(1)	.941	(2)	NA	(6)	.849	(3)	.783	(5)
triazines	.938	(2)	.920	(3)	.905	(4)	.755	(5)	.947	(1)	.677	(6)
waveform21	.770	(5)	.762	(6)	.796	(4)	.869	(1)	.844	(2)	.809	(3)
waveform40	.738	(6)	.741	(5)	.792	(4)	.866	(1)	.834	(2)	.801	(3)
yeast	.526	(6)	.562	(5)	.577	(3)	.589	(2)	.591	(1)	.573	(4)
mean rank	4.05		2.73		3.17		3.74		2.54		4.78	

Later, in Section IV, we also evaluate our method in a more realistic, large-scale study. It uses experiments on 290 algorithm-parameter combinations and 80 UCI datasets, taken from the experiment database ([18],[19]).

The data for the small-scale study is shown in Table I. It presents the accuracies of the different classification algorithms on all prior datasets. Each accuracy figure represents a mean of 10 values obtained in 10-fold cross-validation. The ranks of the algorithms are shown in brackets just after the accuracy value. For instance, if we consider dataset $abalone$, algorithm MLP is attributed rank 1 as its accuracy is highest on this problem. The second rank is occupied by $LogD$ etc.

To construct an initial ranking of all algorithms, we rank all algorithms over all datasets. This is done by aggregating the individual rank values by calculating the *mean rank* of each algorithm. The mean rank of algorithm aj, represented by R_{aj}, is $\frac{1}{n}\sum_{di=1}^{n} R_{aj,di}$ where $R_{aj,di}$ represents the rank of algorithm aj on dataset di and n represents the number of datasets. This is a quite common procedure, often used in machine learning to assess how a particular algorithm compares to others [20].

In Table I the mean ranks are shown at the bottom of the table. These values permit us to obtain the final ordering of algorithms, O. In our small-scale study, $O = \langle MLP, J48, JRip, LogD, IB1, NB\rangle$.

B. The Most Promising Competitor

The final ordering of algorithms, O, established in the previous step identifies the initial best candidate algorithm, i.e. the one appearing in the first position. Let us represent this algorithm as a_{best}. In our small-scale study, $a_{best} = MLP$. Next, we need to identify its *best competitor*. For this, all algorithms a_k in O are considered (except a_{best} itself). For each algorithm, we analyze the information of past experiments to determine its chances of outperforming a_{best} on the new dataset, see Equation 3. Indeed, we count the number of datasets on which a_k outperforms a_{best}. As such, the most promising competitor, $a_{\text{best-comp}}$, satisfies the following expression :

$$\frac{\arg\max_{a_k} \sum_{dj \in Ds}(i(M(a_k, d_j) \geq M(a_{best}, d_j)))}{|Ds|} \quad (4)$$

Should there be two or more algorithms that have the same probability of outperforming a_{best}, then the competitor ranked highest in O is chosen. That is, we follow a hill-climbing approach.

After the test has been carried out, the losing algorithm is removed from O, and the winner is marked as a_{best}. Furthermore, we also use this test to optimize the process of searching for the most promising competitors.

In the expression above the term Ds represents *similar datasets*. These are initially all datasets, but later, as pairwise tests have been carried out, Ds becomes an ever smaller subset for which certain conditions get verified.

Suppose the test a_i versus a_j was carried out on the new dataset and the result was $a_i > a_j$. Here the shorthand $a_i > a_j$ is used to indicate that a_i outperformed a_j. As discussed in Section II-B, the *similar datasets* are those where the same condition is verified (i.e. $a_i > a_j$). After each iteration of the AT procedure, all datasets violating this condition will be removed from Ds.

C. Conducting Tests on a Pair of Algorithms

Having now identified the best candidate algorithm a_{best} and its most promising competitor $a_{\text{best-comp}}$, we can perform a pairwise test on the new dataset.

We can use any valid classifier evaluation procedure in this step. It could be cross-validation procedure, or some faster method. Here we have opted for a relatively fast meta-learning method based on partial learning curves [9], which predicts which algorithm is better out of the two.

This method is referred to as SAM (Selection of Algorithms using Meta-learning). As this method has been described elsewhere [8], [9], [11], the readers who are familiar with this, can skip this section.

The problem of determining which one of the two given algorithms is better can be regarded as a classification problem. If a_i wins over a_j we obtain one class value (here we use 1). If it loses, we obtain a different class value (here -1). If the result is a draw, we obtain yet another class value (here 0). The SAM method predicts this value using both statistical data characteristics and *partial learning curves*: information about accuracies of the two classifications algorithms on ever larger samples of the data for different datasets.

The performance of the two algorithms on a set of samples sheds some light on the final outcome of the learning process. Fig. 1 illustrates this. Its shows the measured accuracy on three samples of data s_1, s_2 and s_3. Here $s_1 \cdots s_m$ are used to represent random samples of increasing size.

The two curves have been extended on the basis of other similar curves retrieved from the meta-database. It enables to determine that algorithm A_q will achieve higher accuracy at the end, that is, when the full set of examples has been used.

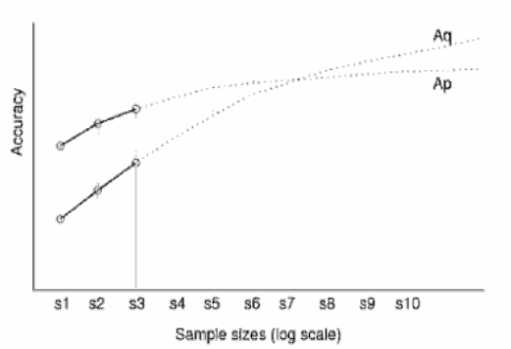

Figure 1. Example of two extended learning curves determining which algorithm is better

The method SAM is divided in two subroutines: SAM_F and $SetGen$. Subroutine SAM_F predicts which algorithms is better using a fixed set of *meta-features* indicated as parameters. These meta-features represent both general data characteristics like *number of cases*, *class entropy*, etc. and also known points of *partial learning curves*.

Subroutine $SetGen$ searches for a subset of the best meta-features satisfying two criteria: (1) maximize the improvement the performance SAM_F (accuracy in deciding which of the two algorithms is better), while (2) trying to reduce the costs involved in obtaining the values of meta-features (computing a point on the learning curve requires conducting a test, i.e. initiating learning and evaluating the model). Therefore we attend to both criteria.

This method also exploits the active testing strategy discussed in the previous subsection.

Subroutine SAM_F requires as input a fixed set of samples for each algorithm. Let S_i (S_j) represent the samples required to characterize algorithm a_i (a_j). So, for instance, the samples passed as input can be $S_i =< s_1, s_2, s_3 >$ and

$S_j =< s_1, s_2 >$. The method encompasses the following steps:

1) Compute the data characteristics for the new dataset d.
2) Characterize the new dataset d by conducting experiments with algorithm a_i on S_i samples and measuring accuracies. Repeat this for a_j on S_j samples. In other words build two partial learning curves.
3) Compute the distances between the information relative to dataset d and stored information relative to all other datasets $d_1 \cdots d_n$.
4) Identify the subset of k nearest datasets.
5) For each of the k nearest datasets identified and retrieved in the previous step, *adapt* each pair of learning curves to the new partial learning curves build for dataset d. Adaptation is done by rescaling each retrieved learning curve in order to minimize the square distance from this curve to the respective partial learning curve for dataset d.
6) For each pair of adapted curves decide which algorithm achieves higher performance on the adapted and extended learning curves.
7) Identify the algorithm that is better on the new dataset d, by considering the results on k pairs of nearest datasets.

Method SAM was evaluated using the same leave-one-out methodology discussed in Section IV. The results have shown that the method was about 90% accurate in predicting the best algorithm. The mean runtime of the method when compared to selection using cross-validation was about 7 times lower.

D. Repeating the Method and Stopping Criteria

The whole process of identifying the best competitor (step 2) and conducting tests (step 3) is repeated until no competitor can be identified. There is a threshold value on the probability that the competitor outperforms the best one. If no competitor passes this threshold, the procedure stops. The method outputs the algorithm that has successfully defended itself against all competitors. This algorithm can be regarded as the overall winner.

IV. Evaluation and Results

This section evaluates the AT procedure. First, we continue our small-scale (toy) study in Section IV-A, and in Section IV-B we evaluate our method in a more realistic, large-scale study.

In both cases the proposed method was evaluated using a *leave-one-out* method [22]. The experiments reported here involve N datasets and so the whole procedure was repeated N times. In each cycle one dataset was left out for testing (all meta-data for that dataset was removed), and the remaining $N-1$ datasets were used to determine the best candidate algorithm.

Table II
Rank of predictions / real / default

Dataset	test AT_{full}	rank	test $AT_{fast/full}$	rank	test AT_{fast}	rank	rank default	true best
abalone	MLP	1	MLP	1	MLP	1	1	MLP
acetylation	IB1	1	J48	2	J48	2	6	IB1
adult_metal	J48	1	JRIp	3	JRIp	3	5	J48
allbp	JRIp	1	J48	2	J48	2	3	JRIp
allhyper	J48	1	JRIp	2	JRIp	2	3	J48
ann	J48	1	J48	1	J48	1	3	J48
byzantine	MLP	1	IB1	2	IB1	2	1	MLP
car	MLP	1	MLP	1	MLP	1	1	MLP
cmc	MLP	1	J48	2	J48	2	1	MLP
contraceptive	MLP	1	J48	2	J48	2	1	MLP
injury_severity	MLP	1	J48	2	J48	2	1	MLP
internetad	JRIp	1	JRIp	1	JRIp	1	6	JRIp
isolet	MLP	1	MLP	1	NB	3	1	MLP
krkopt	MLP	1	LogD	3	LogD	3	1	MLP
krvskp	MLP	1	J48	2	J48	2	1	MLP
led24	J48	3	J48	3	J48	3	5	LogD
letter	IB1	1	MLP	4	MLP	4	4	IB1
mfeat	MLP	1	IB1	2	IB1	2	1	MLP
musk	MLP	1	MLP	1	MLP	1	1	MLP
nursery	MLP	1	MLP	1	MLP	1	1	MLP
optdigits	IB1	1	IB1	1	IB1	1	2	IB1
page	JRIp	1	JRIp	1	JRIp	1	4	JRIp
parity	MLP	1	MLP	1	MLP	1	1	MLP
pendigits	JRIp	2	IB1	1	IB1	1	5	IB1
pyrimidines	MLP	1	MLP	1	MLP	1	1	MLP
quadrupeds	MLP	3	IB1	3	IB1	3	3	IB1
quisclas	JRIp	2	JRIp	2	JRIp	2	4	LogD
reclJan2jun97	MLP	1	LogD	2	LogD	2	1	MLP
sat	IB1	1	IB1	1	IB1	1	2	IB1
segmentation	IB1	1	IB1	1	IB1	1	4	IB1
shuttle	J48	1	JRIp	2	JRIp	2	4	J48
sick	J48	1	J48	1	J48	1	3	J48
spambase	J48	1	J48	1	J48	1	5	J48
splice	MLP	1	MLP	1	MLP	1	1	MLP
taska_part_hhold	J48	1	JRIp	2	LogD	6	5.5	J48
thyroid0387	J48	1	J48	1	NB	5	3	J48
triazines	MLP	1	J48	3	J48	3	1	MLP
waveform21	LogD	1	LogD	1	LogD	1	2	LogD
waveform40	LogD	1	LogD	1	LogD	1	2	LogD
yeast	MLP	1	LogD	2	LogD	2	1	MLP
Mean Rank		1.15		1.68		1.92	2.54	

A. Small-scale Study

1) Experimental set-up: Our small-scale study involves the 6 classification algorithms and 40 datasets mentioned in Section III-A. As we know what the actual best algorithm is on each test dataset, we can use the actual rank of the algorithm on the test dataset as a measure of success. Ideally, we should predict the algorithm ranked first on that dataset.

To improve upon the 'default prediction' of the best overall algorithm, MLP, with a mean rank of 2.54 (see Table I), the mean rank of all N predictions should be lower than that value. In case we always predict the best algorithm, the mean rank will be 1.

We denote the full AT procedure described above as AT_{full}. We also include two slower variants, which differ in the execution of the pairwise tests. First, instead of our meta-learning approach (SAM), we can simply use a (slower) cross-validation (CV) procedure for each pairwise test, referred to as AT_{full}. Second, the SAM method used in AT_{full} has a time limit. In 5 of the 40 dataset, a time-out occurs, in which case the dataset is skipped in the competitor selection procedure. Therefore, we also include a hybrid $AT_{fast/full}$ variant, which performs a cross-validation procedure in those cases.

2) Results: The results of our method and the two variants are shown in Table II. It can be seen that the mean rank of the fast method, AT_{fast}, is 1.92. This is a

24% improvement over the default prediction of the best-performing algorithm in our small-scale experiments, which is MLP with a mean rank of 2.54.

The method referred to as $AT_{fast/full}$ in Table II achieves an even better mean rank (1.68), a 34% improvement of the mean rank. This is not surprising, as in all situations when SAM did not return a result we have conducted a CV test.

Finally, the full cross-validation testing, AT_{full} in Table II, achieved a mean rank of 1.15 representing a 55% improvement of the mean rank. This is an excellent result as it is very near to the ideal value.

3) Analysis of the number of steps needed: Let us analyze the runtimes of the fast method (AT_{fast}) proposed here. Our method is many times faster than the method that would perform all possible pairwise tests in a CV procedure to select the best algorithm. The first saving comes from using the active testing strategy. In our set-up with 6 algorithms, the number of all possible pairwise tests is $\binom{6}{2} = 6 * 5/2 = 15$. When looking at the average number of tests needed to reach a decision in all N cases, we learn that it typically takes no more than 3 tests, $1/5$ of all possible tests. Furthermore, the fast SAM tests are also about 7 times faster than tests that involve CV [9].

If we were to use CV to select the best algorithm (identified as *true best* in Table II) for the new dataset, we would of course not need to run pairwise tests, but simply run CV for the 6 algorithms. Still, the time saving is quite substantial.

Also, the method referred to as AT_{full} in Table II achieved a very good mean rank in small-scale experiments, but we need to be aware that this method is not so fast. It requires 3 CV tests, each involving 2 algorithms. As one of the algorithms is common in two subsequent tests, we need to evaluate 4 algorithms using CV, instead of 6. Still, as the number of tests grows approximately with $log(|A|)$, where $|A|$ represents the number of algorithms (and their different variants), the time savings becomes ever greater as more algorithms are considered. The large-scale study below confirms this.

4) Contigency Plan: Suppose the method presented here is used with all datasets and all recommended tests get recorded. As some of tests can then be merged (here we have used a manual process) the resulting structure appears in the form of a tree. The resulting tree is shown in Fig. 2. However, it would not be difficult to automate fully this process.

We note that the tree starts with the most important test to be carried out which in our case is MLP $versus$ $J48$. The result of this test determines what needs to be done next. Note that we do not know beforehand what the result will be. Still the figure contemplates both alternatives (as both were used in the past). This is why we refer to the tree as a *contingency plan*. It provides a plan regard what should be done in different situations (e.g. when $MLP < J48$ etc.).

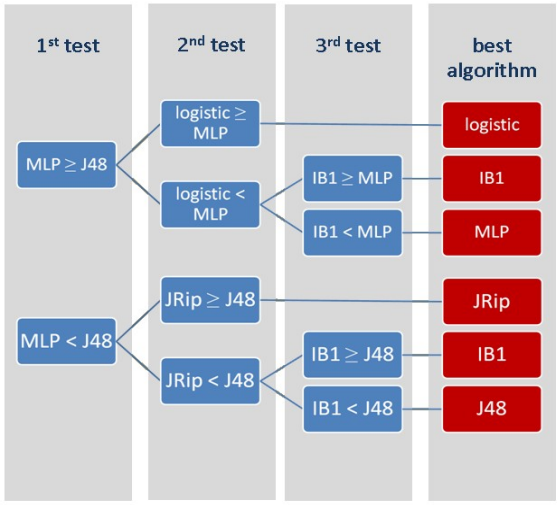

Figure 2. Contigency Plan

Note that the contingency plan exploits metaknowledge, but reorganizes it in the form of an explicit plan that can be directly followed in a new situation either by the user or by the system.

The contingency plan discussed here could be seen also as a kind of meta-model. As we do not know how accurate it is, we do not suggest that is should be used in substitution of our AT algorithm. Rather, it can serve as a nice visualization tool.

5) Discussion: As the method presented here is based on mean ranks, it is interesting to analyze the result of Bonferroni-Dunn test [20] which is an appropriate statistical test when multiple algorithms are being compared on different datasets. The result of this test is shown in Fig. 3. The length of each horizontal blue line shows the critical distance (CD) for $p = 0.1$. The analysis of this figure confirms that the best algorithm is indeed MLP and that the best competitor in the first step is $J48$. The large overlap of the critical distances of MLP and J48 suggests that we cannot easily decide which algorithm is a priori better. In other words, any the decision that could be made is rather uncertain. This is a good reason for carrying out a test. Of course, the situation depends very much on which datasets are considered (i.e. which datasets are included in Ds).

B. Large-scale Study

1) Experimental set-up: This study involved 290 algorithm-parameter combinations, which were extracted from the experiment database [18], [19]. This set included many different algorithms from the Weka platform [16], and a selection of popular algorithms where also varied by assigning different values to their most important parameters. These include SMO (a support vector machine (SVM) trainer), MultilayerPerceptron, J48 (C4.5), 1R, RandomForest, Bagging and Boosting. Moreover, different SVM kernels

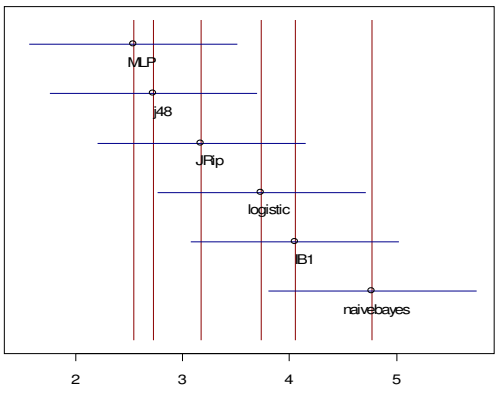

Figure 3. Results of Bonferroni-Dunn test relative to the initial situation

were used with their own parameter ranges, and all learners were used as base-learners for ensemble learners. The 80 datasets used in this study were all from UCI [17].

2) Results: In this study our aim was to examine the rank of the predicted algorithm. As we have identified that there were some outliers, we have calculated a median value, rather than the mean that would be distorted by them. The median rank of our recommendation was about 20, while the mean rank was about 42. If we compare it to the mean rank of the best algorithm overall - $Bagging.I.75..100PART$, whose mean rank was 66, we see that we got quite good reduction.

3) Analysis of the number of steps needed: As the number of algorithms in this study is 290, we expect that the number of tests needed would be around $log_2(290) = 8.2$. We have conducted a study to determine the real number of tests. These varied for different datasets between 5 and 10. The values 6 and 7 were the most frequent values.

It is interesting to analyze the steps carried out by our method in the course of identifying the right algorithm. Let us analyze this in more detail.

Let us consider, for instance, the first dataset in our study, which is $abalone$. The system start with the best performing algorithm overall mentioned above, i.e. $Bagging.I.75..100PART$ (rank 1 in our list, but with mean rank 66). The system then conducts a test that involves $Bagging.25..50.LMT$ which has rank 2 in the list (but with mean rank 66,4). The competitor wins in the test and so another competitor is identified, $Bagging.50..75.LMT$, that is again a winner. This algorithm is challenged by $Bagging.50..75.SimpleLogistic$ ranked 34 in our list. This algorithm defends itself in two further runs, but finally looses against $Bagging.1..25.MultilayerPerceptron$ (rank 19 in our list) that is the overall winner on this dataset. In total 8 test were needed to identify the presumed winner.

V. Conclusions

In this paper we have discussed the problem of selecting the best classification algorithm for a specific task. We have suggested a method referred to as *active testing*. Our aim is to reduce the number of test by carefully selecting which tests should be carried out.

The method described (AT_{fast}) uses meta-knowledge concerning past experiments and proceeds in an iterative manner. This enables us to identify not only potentially best algorithm, but also its best competitor. This determines the test to be carried out which determines which algorithm should be eliminated from consideration.

The method was evaluated in a leave-one-out fashion. The results show that the method is indeed effective in determining a well-ranked algorithm using a limited number of tests.

We have used two different set-ups to evaluate the method. The first one was a small-scale study, which included 6 classification algorithms and 40 datasets. This study was used to illustrate how the mehtod works. The mean rank of the fast method was 1.92, which we consider to be a good result, as it is much better than the mean rank of the best performing algorithm, that is, MLP with a mean rank of 2.54. A somewhat slower variant uses a slower test based on cross-validation. This variant is referred to as AT_{full}. The mean rank of this method is 1.15 which is very near to the ideal value.

The second set-up was a large-scale study. It included 290 algorithm-parameter combinations, which were extracted from the experiment database [18], [19]. This set included many different algorithms from the Weka platform [16] and their variants that were obtained by assigning different values to their most important parameters.

The median rank of our recommendation was about 20, while the mean rank was about 42. If we compare it to the mean rank of the best algorithm overall ($Bagging.I.75..100PART$), whose mean rank was 66, we see that we got quite good reduction.

We have conducted a study to determine the real number of tests. Here we could verify that our method has lead to very significant savings. In many cases only 6 or 7 tests were sufficient to identify a very competitive algorithm. We consider that this is an important result which may be of use to others.

References

[1] D. Michie, D.J.Spiegelhalter, and C.C.Taylor, *Machine Learning, Neural and Statistical Classification.* Ellis Horwood, 1994.

[2] P. Brazdil, C. Soares, and J. Costa, "Ranking learning algorithms: Using IBL and meta-learning on accuracy and time results," *Machine Learning*, vol. 50, pp. 251–277, 2003.

[3] R. Vilalta and Y. Drissi, "A perspective view and survey of meta-learning," *Artif. Intell. Rev.*, vol. 18, no. 2, pp. 77–95, 2002.

[4] K. A. Smith-Miles, "Cross-disciplinary perspectives on meta-learning for algorithm selection," *ACM Comput. Surv.*, vol. 41, no. 1, pp. 1–25, 2008.

[5] J. R. Rice, "The algorithm selection problem," ser. Advances in Computers, M. Rubinoff and M. C. Yovits, Eds. Elsevier, 1976, vol. 15, pp. 65 – 118. [Online]. Available: http://www.sciencedirect.com/science/article/B7RNF-4S8V0RY-7/2/be446024f53455be2bca3d05b47087c1

[6] C. Soares, J. Petrak, and P. Brazdil, "Sampling-based relative landmarks: Systematically test-driving algorithms before choosing," in *Proceedings of the 10th Portuguese Conference on Artificial Intelligence (EPIA 2001)*. Springer, 2001, pp. 88–94.

[7] J. Fürnkranz and J. Petrak, "An evaluation of landmarking variants," in *Proceedings of the ECML/PKDD Workshop on Integrating Aspects of Data Mining, Decision Support and Meta-Learning (IDDM-2001)*. Springer, 2001, pp. 57–68.

[8] R. Leite and P. Brazdil, "Predicting relative performance of classifiers from samples," in *ICML '05: Proceedings of the 22nd international conference on Machine learning*. New York, NY, USA: ACM Press, 2005, pp. 497–503.

[9] ——, "An iterative process for building learning curves and predicting relative performance of classifiers," in *Progress In Artificial Intelligence, Proceedings of the 13th Portuguese Conference on Artificial Intelligence Workshops (EPIA 2007)*, ser. Lecture Notes in Computer Science, J.Neves, M.F.Santos, and J.Machado, Eds., vol. 4874. Guimarães, Portugal: Springer, December 2007, pp. 87–98. [Online]. Available: http://www.liaad.up.pt/pub/2007/LB07a

[10] A.Kalousis and M.Hilario, "Feature selection for metalearning," in *D.Cheung et al.(eds): Proc. of the Fifth Pacific–Asia Conf. on Knowledge Discovery and Data Mining*. Springer, 2001.

[11] R. Leite and P. Brazdil, "Active testing strategy to predict the best classification algorithm via sampling and metalearning," in *Proceedings of the 19th European Conference on Artificial Intelligence - ECAI 2010 (to appear)*, 2010.

[12] V. Fedorov, *Theory of Optimal Experiments*. Academic Press, New York, 1972.

[13] A. McCallum and K. Nigam, "Employing EM and pool-based active learning for text classification," in *Proceedings of the 5th Int. Conf. on Machine Learning*, 1998, pp. 359–367.

[14] B. Long, O. Chapelle, Y. Zhang, Y. Chang, Z. Zheng, and B. Tseng, "Active learning for rankings through expected loss optimization," in *Proceedings of the SIGIR'10*. ACM, 2010.

[15] K. De Grave, J. Ramon, and L. De Raedt, "Active learning for primary drug screening," in *Proceedings of Discovery Science*. Springer, 2008.

[16] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: an update," *SIGKDD Explor. Newsl.*, vol. 11, no. 1, pp. 10–18, 2009. [Online]. Available: http://dx.doi.org/10.1145/1656274.1656278

[17] C. Blake and C. Merz, "UCI repository of machine learning databases. http://www.ics.uci.edu/∼mlearn/mlrepository.html," 1998. [Online]. Available: http://www.ics.uci.edu/∼mlearn/MLRepository.html

[18] H.Blockeel, "Experiment databases: A novel methodology for experimental research," in *Leacture Notes on Computer Science 3933*. Springer, 2006.

[19] J. Vanschoren and H. Blockeel, "A community-based platform for machine learning experimentation," in *Machine Learning and Knowledge Discovery in Databases, European Conference, ECML PKDD 2009*, vol. LNCS 5782. Springer, 2009, pp. 750–754.

[20] J. Demsar, "Statistical comparisons of classifiers over multiple data sets," *The Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.

[21] Y. Freund, H. Seung, E. Shamir, and N. Tishby, "Selective sampling using the query by committee algorithm," *Machine Learning*, vol. 28, pp. 133–168, 1997.

[22] T. M. Mitchell, *Machine Learning*. New York: McGraw-Hill, 1997.